IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF TEXAS
TYLER DIVISION

SYBASE, INC., **Plaintiff,** **v.** **VERTICA SYSTEMS, INC.,** **Defendant.**	**Civil Action No. 6:08-cv-24 LED**

DEFENDANT VERTICA SYSTEMS, INC.'S CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

TABLE OF AUTHORITIES

CASES

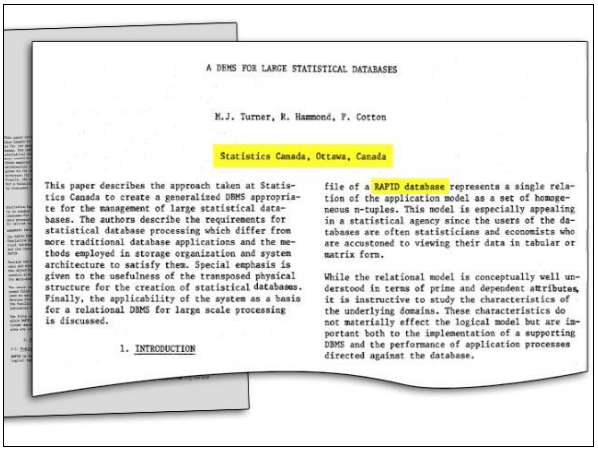
STATUTES

I. INTRODUCTION

Sybase did not invent column-based storage.

Sybase would have this Court believe that the "pioneering" "invention" of the '229 patent[1] was storing data from a database table by column instead of by row—and that this alleged brainchild occurred in 1995. Sybase could not have invented column-based storage in 1995, because column-based storage was already in use in various statistical databases by the 1970s.

One such column-based storage database was described in Vertica's Technology Tutorial and was called the RAPID ("Relational Access Processor for Integrated Databases") System[2]:



A DBMS FOR LARGE STATISTICAL DATABASES

M.J. Turner, R. Hammond, P. Cotton

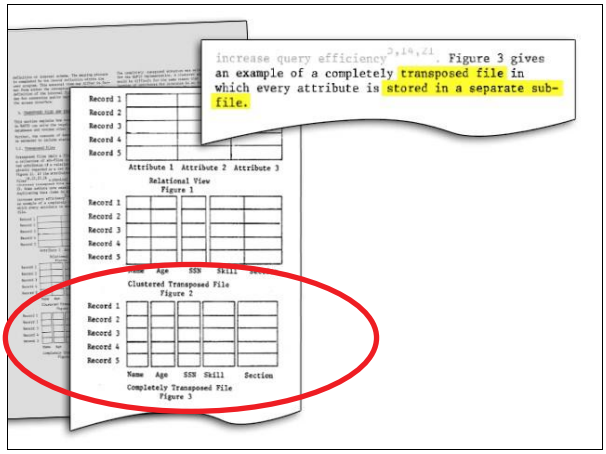
Statistics Canada, Ottawa, Canada

This paper describes the approach taken at Statistics Canada to create a generalized DBMS appropriate for the management of large statistical databases. The authors describe the requirements for statistical database processing which differ from more traditional database applications and the methods employed in storage organization and system architecture to satisfy them. Special emphasis is given to the usefulness of the transposed physical structure for the creation of statistical databases. Finally, the applicability of the system as a basis for a relational DBMS for large scale processing is discussed.

1. INTRODUCTION

file of a RAPID database represents a single relation of the application model as a set of homogeneous n-tuples. This model is especially appealing in a statistical agency since the users of the databases are often statisticians and economists who are accustomed to viewing their data in tabular or matrix form.

While the relational model is conceptually well understood in terms of prime and dependent attributes, it is instructive to study the characteristics of the underlying domains. These characteristics do not materially effect the logical model but are important both to the implementation of a supporting DBMS and the performance of application processes directed against the database.



increase query efficiency[2,14,21]. Figure 3 gives an example of a completely transposed file in which every attribute is stored in a separate sub-file.

Record 1
Record 2
Record 3
Record 4
Record 5
Attribute 1 Attribute 2 Attribute 3
Relational View
Figure 1

Clustered Transposed File
Figure 2

Name Age SSN Skill Section
Completely Transposed File
Figure 3

The RAPID system was developed by Statistics Canada and was used to process the 1976 Census of the Population of Canada. By 1981, the RAPID system had also been installed in various government agencies in Canada, United States, Sweden, Brazil, East Germany, and Hungary. RAPID was also installed by the United Nations at its headquarters in Chili and Costa

[1] U.S. Patent No. 5,794,229 ("the '229 patent") is attached hereto as Exhibit A.

[2] These screen shots display the cover page of the seminal Turner, Hammond, Cotton article discussing the RAPID system and page illustrating the column-based storage, which RAPID called a "completely transposed" database and the '229 patent calls a "vertically partitioned" database. The bottom figure on the right-most page illustrates how each column of a database table is stored separately in separate "sub-files."

Rica. (Ex. B, Schmidt & Brodie, Relational Database Systems: Analysis and Comparison (Springer-Verlag 1983), pp. 442-43, 476). As discussed below, the RAPID system was also well-documented in various articles and manuals in the 1970s and 1980s, including in the seminal work by Turner, Hammond, and Cotton, entitled "A DBMS for large Statistical Databases." (Ex. C).[3]

Because the RAPID system used column-based storage decades before the '229 patent, the claims of the '229 patent must cover something less than column-based storage. In fact, the claims of the '229 patent cover **one specific way** to implement column-based storage—through the use of something called "page chains." The construction of the term "page chain" is a central issue in this claim construction proceeding.

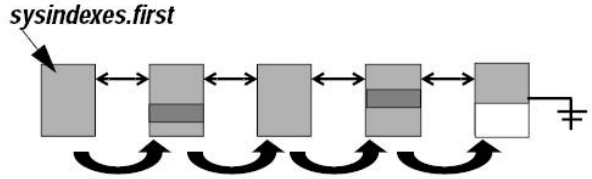
Sybase is trying to use the claim construction process to turn its fairly narrow patent—limited to the use of "page chains"—into something much broader so that it can maintain its infringement case against Vertica. Vertica, like the prior art RAPID system, does not use "page chains" according to the ordinary meaning of that term. Neither RAPID nor Vertica place a reference **in the data page itself** pointing to the next page in a chain. Instead, both RAPID and the Vertica system locate pages using directories and indexes stored **outside** the data pages. The pages in RAPID and Vertica are thus not linked together to form a page chain.

The following illustrations (from Sybase's own literature) visually distinguish page chains from a page index. On the left is a "page chain" which has a series of pages with links in the pages for jumping from one page to the next. On the right, an "allocation page" is an index that points to various pages which are **not** linked together in a chain. The Sybase technical manual containing these figures states that the pages on the left are "**linked** into a doubly-linked

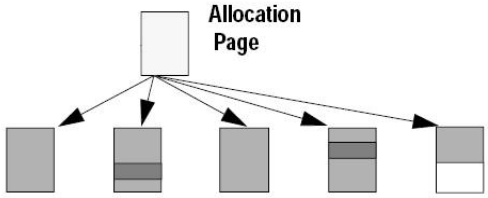
[3] The full bibliographic citation is Turner, Hammond, Cotton, "A DBMS for Large Statistical Databases," Proceedings of VLDB 1979, Rio de Janeiro, Brazil, pp. 319-327.

list of pages **by pointers on each page**" and that "**each page stores a pointer to the next page in the chain** and to the previous page in the chain." (Ex. D, Sybase Adaptive Server Performance And Tuning Guide, pp. 3-11 to 3-14). That same manual states that the collection of pages on the right which are "associated" by an index in the Allocation Page "are **not linked into a page chain**."

Example of "Page Chain"



Example of Index Which Is Not Linked Into A Page Chain



Sybase's infringement claim hinges on eviscerating the "page chain" limitation found in every claim of the '229 patent.

The "page chain" limitation is not the only term Sybase is trying to "read out" of the '229 patent. Repeatedly, Sybase's claim constructions take technical terms that have specific meanings in the database field and generalize them into broad concepts having virtually no boundaries[4]:

- "page" expands to "logical structure";
- "linking" expands to "associating";
- "chain" expands to "collection";

[4] Oddly, Sybase makes one significant departure from its approach of seeking broad and vacuous constructions. It seeks to recast "computer system" (a term requiring no construction) as "relational database management system." Vertica addresses this claim term in detail later in this brief, but even without argument, the Court no doubt recognizes that the term "computer system" does not mean "relational database management system" and that Sybase's construction is prima facie incorrect.

- "header" expands to "associated metadata."

Sybase's vague constructions undermine the very purpose of claims—which is to define the metes and bounds of Sybase's right to exclude. *See Halliburton Energy Services, Inc. v. M-I LLC*, 514 F.3d 1244, 1249 (Fed. Cir. 2008). The Court should construe these terms according to their ordinary and customary meaning in the database field. Vertica respectfully requests that the Court adopt its proposed constructions.

Vertica suggests that the Court review the Technology Tutorial submitted by Vertica Systems, Inc. on September 24, 2009 for a primer on many of the technical concepts to be discussed in this Brief. Another copy of Vertica's Technology Tutorial is also included with the courtesy copies of this brief being delivered to chambers.

Also, there are two companion Motions being filed along with this Claim Construction Brief and both have a direct bearing on claim construction:

1. DEFENDANT VERTICA SYSTEMS, INC.'S MOTION FOR SUMMARY JUDGMENT ON INDEFINITENESS

2. DEFENDANT VERTICA SYSTEM, INC.'S MOTION FOR SUMMARY JUDGMENT OF ANTICIPATION OF CLAIM 1 BASED ON SYBASE'S PROPOSED CONSTRUCTION OF "PAGE CHAIN"

II. BACKGROUND

A. Column Storage Was Invented Decades Before Sybase Filed Its Patent Application.

Sybase opens its Claim Construction Brief touting the alleged "invention" without mentioning the requirement that the data must be stored in "page chains." In fact, Sybase downplays the importance of this critical limitation throughout its brief. Here are Sybase's first two sentences:

> Sybase's pioneering U.S. Patent No. 5,794,229 ('the '229 Patent') literally turned the database industry on end through its novel invention of storing all the columns of a database table by *vertically partitioning* the data instead of storing it

> horizontally (which has been the accepted norm in the relational database field for 25 years). By storing data in a column-wise fashion, the claimed system and method of the '229 Patent can process user queries with far greater speed and efficiency.

(Dkt. No. 107, Sybase Claim Construction Brief, p.1)[5]. Sybase appears to assert that it invented column-based storage—that it was the first to store columns by "vertically partitioning" the data in a database table (that is, separating each column of the database table).[6]

Contrary to Sybase's assertion, vertical partitioning (that is, column-based storage) was already in widespread use in statistical databases by the alleged "invention" date of 1995. These prior art systems stored columns separately for the same reasons described in the '229 patent: It is much more efficient to store the columns separately if only a few columns of data need to be read from disk (as is typical in statistical analysis).

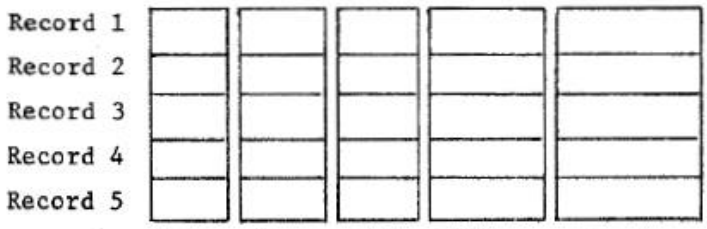
The Statistics Canada RAPID system is one of the statistical database that used column-based storage before the '229 patent.[7] As described in the 1979 paper by Turner, Hammond, and Cotton, The RAPID system was a "completely transposed file in which every attribute is stored in a separate sub-file." (Ex. C, p. 321). The storage of each column separately from any other column (which can be called either a "vertically partitioned" file or a "completely transposed" file) is clearly illustrated in the article by the following figure:

---

[5] Emphasis added throughout this brief unless otherwise indicated.

[6] The parties agree that "vertically partitioning" the data means separating each column of a database table. Each column is separated specifically so that one column can be read from storage without having to read the other columns. The official agreed construction is: "Separating all data in each column in the database table from any data from any other column in the database table, such that all data in each column can be read from storage without reading any data from any other column."

7 The RAPID system is described in Vertica's tutorial.



Completely Transposed File
Figure 3

The article explains that each attribute (such as name, age, social security number, skill, and section) is stored separately in separate sub-files. The article even discloses the same two rationales for vertically partitioning the data as disclosed in the '229 patent—to avoid having to bring in the non-selected columns from disk when executing a statistical query and to compress the relatively uniform attribute (*i.e.*, column) data:

> [S]tatistical queries often request a large percentage of the tuples [rows] in a relation. RAPID, as a completely transposed system, supports this requirement in two ways. First, when a query must retrieve many tuples for one or more attributes [columns] its required I/O activity and data transfer is independent of the size of the non-selected attributes. The second reason is related to the fact that RAPID stores in each sub-file elements of a single attribute whose storage requirements are identical. For qualitative attributes with a small range, a small number of bits may be used to store each element.

(*compare* Ex. C, p. 321; *with* Dkt. No. 107, Sybase's Claim Construction Brief, pp. 5-6 and column 3 of the '229 patent).

RAPID was not even the first database to use column-based storage. The 1979 article explains that "RAPID had two direct predecessors which both used a completely transposed file structure [i.e., were vertically partitioned]. The file system used to process the retrieval part of the 1971 Canadian Housing and Population Census used transposed files where each attribute was stored in a separate direct access file." (Ex. C, p. 322)

Sybase's assertion that it invented column-based storage is disingenuous and dead wrong. Vertica informed Sybase of the RAPID system four months after Sybase filed the lawsuit. (Ex.

E, April 28, 2008 E-mail to Mr. Nathan at Sybase attaching Turner reference). It is impossible to square the truth—that statistical relational databases were vertically partitioned from at least the 1970s—with Sybase's assertion in its claim construction brief that Sybase's '229 patent was "pioneering" because it stored all columns of a database table by column instead of by row. Sybase also makes the same incredible—and false—assertion in its Technology Tutorial.

In construing the terms of the '229 patent, the Court must be mindful that—at best—the '229 patent represents a modest change from then then-existing column-based storage databases that did not use "page chains" as that term is correctly—and narrowly—construed. If this Court were to accept Sybase's proposal and interpret a "page chain" to be merely a "collection" of "associated" "logical structures" holding the column data, then the patent would be anticipated.

For example, claim 1 and many of its dependent claims would be anticipated by the prior art Turner, Hammond, and Cotton article describing the RAPID system (Ex. C). The remaining claims would be invalidated by the collection of user manuals describing the RAPID system. Thus, the Court should adopt Vertica's construction in order to avoid certain anticipation of at least the broad claims. *See Phillips v. AWH*, 415 F.3d 1303, 1327 (Fed. Cir. 2005) (as a "last resort," claim terms should be construed to preserve the validity of the claims). To back up this assertion, Vertica is filing with this Brief a Motion For Summary Judgment of Anticipation of Claim 1 Based On Sybase's Proposed Construction of "Page Chain."

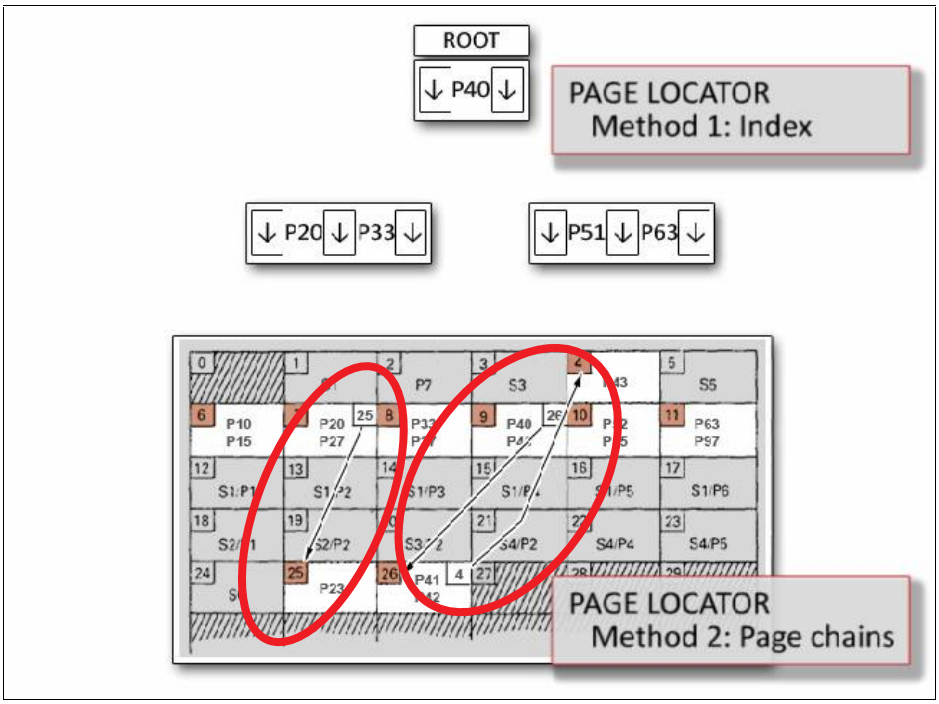
B. The Prosecution History Supports A Narrow Construction of Page Chain

The Patent Office originally rejected Sybase's claims finding that each element was found in the Naecker and Snellen references.[8] (Ex. F, Office Action dated July 18, 1997, p. 5, ¶ 11). The Examiner stated that the prior art Naecker reference—**which disclosed B-Tree**

[8] Sybase discusses the prosecution history surrounding claims dropped from prosecution due to an "obviousness-type double patenting" rejection. That commentary has nothing to do with the claims that actually issued in the '229 patent.

indexes and catalogs[9]—disclosed everything but "linked" pages: "Naecker does not explicitly indicate linking the pages which store the data of a particular column." (*Id.* at p. 6, ¶ 14).

The Examiner found the linking element in the Snellen reference. (*Id.* at p. 6, ¶ 16) Snellen (like the '229 patent itself) teaches the use of pointers to link pages: "When new records exceed the allocated data page space, the ISAM algorithm creates overflow pages to hold the records. **A pointer in the original page space indicates the overflow pages**." (Ex. G, David Snellen, "Ingres Table Structures", DBMS, vol. 5. no. 8, p. 60(3), July 1992, p. 2). The use of pointers in an ISAM system to handle overflow pages is described in Vertica's Technology Tutorial provided to the Court:



[9] For example, the Naecker reference states: "The catalog describes the data stored in the database" (VERT0021094) and "RDBMSs now support B-Trees and . . . support multiple indexes on the same field or fields." (VERT0021099) (Ex. G to Sybase's Claim Construction Brief).

As shown in this figure and described in Snellen, a page pointer in one page points to the location of the next overflow page. The pages joined by pointers from one page to the next are **linked**.

If Sybase believes that B-Tree indexes and system catalogs meet the "linking" limitation, it should have "corrected" the Examiner's "mistake" and informed the Patent Office that the Naecker reference does disclose the "linking" limitation and that Snellen was unnecessary. It did not. Sybase knew at the time that it could obtain its claims by relying on the Examiner's narrow (and correct) interpretation of "linking," while a broad "interpretation" would obviate the need to rely on the Snellen reference and preclude allowance of the claims.

Instead, Sybase responded to the Patent Office's rejection by ignoring Naecker and arguing that Snellen merely disclosed the "common use of overflow pages." (Ex. H, Response to Office Action dated January 16, 1998, p. 9). Presumably, Sybase distinguished Snellen on the basis that the patent requires all pages to use links whereas Snellen only required page pointers in overflow situations. Significantly, Sybase did not challenge the Examiner's statement that the Naecker's B-Tree indexes and system catalogs did not disclose the linking of pages.

Further narrowing its claims, Sybase also distinguished the Naecker reference on the basis that the claims required vertical partitioning "irrespective of the storage devices available to the system." Sybase argued that the Naecker reference implemented vertical partitioning by placing small columns on a small, fast disk, and large columns "might be placed on slower disks." (*Id.* at p. 8). The applicant stated: "In contrast to the approach described by Naecker, the approach of the present invention is to break-up the data up vertically regardless of the disk configuration. . . . Quite simply, the disk configuration is not important to the operation of the present invention." (*Id.* (emphasis in original)).

These statements by Sybase to the Patent Office clarify the scope of these claim terms.

III. THE LAW OF CLAIM CONSTRUCTION

This Court is already familiar with the law of claim construction and Vertica will spare the Court a lengthy exegesis. As the Court will see, the terms in question are all common technical terms in the database field. They all have ordinary and customary meanings in the art. Vertica's proposed constructions capture that ordinary and customary meaning based on the entire intrinsic record: the claims, the specification, and the prosecution history, including prior art cited in the prosecution history. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1314 (Fed. Cir. 2005) (en banc).

Vertica then offers extrinsic evidence that further **confirms** the truth and accuracy of Vertica's proposed constructions that are based on the intrinsic record. The terms in question have ordinary and customary meanings in the field, and the extrinsic evidence demonstrates that wholly apart from the patent itself, there was a common usage of the terms in question. *Id.* at 1318.

Two pieces of extrinsic evidence are particularly significant. Both confirm that Vertica has proposed the ordinary meaning and correct construction of the term "page chain." Both confirm that Sybase's construction is too broad and sweeps in technologies that are not "page chains." The first is the deposition testimony of Peter White, one of the two named inventors on the '229 patent. Mr. White testified that a "page chain" is exactly what Vertica says it is. Mr. White specifically distinguishes "page chains" from hierarchical indexes, such as B-Trees. (Ex. I, Deposition of Peter White, p. 40, l. 21 to p. 44, l. 2). Mr. White's testimony is significant and relevant because he was **not** testifying about "what the claims mean." The cases are clear that such testimony is suspect. Rather, he was testifying about what a page chain was in the database field wholly apart from his patent. The Court may view an excerpt of Mr. White's testimony

regarding linking pages into page chains in Vertica's Technology Tutorial (scroll to 17 minutes and 9 seconds).

Mr. White's testimony is thus relevant—and highly credible—expert testimony on the ordinary meaning of the term. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1317 (Fed. Cir. 2005) (en banc); *Howmedica Osteonics Corp. v. Wright Medical Technology, Inc.*, 540 F.3d 1337, 1347 n.5 (Fed. Cir. 2008); *CCS Fitness, Inc. v. Brunswick Corp.*, 288 F.3d 1359, 1368 (Fed. Cir. 2002).

The second significant piece of extrinsic evidence is Sybase's own technical literature which also confirms that Vertica's definition of page chain is correct and that Sybase's definition too broad. That literature, a "Performance and Tuning Guide" discussed in detail later in this brief, is important because it is a public statement by Sybase that was made in the ordinary course of business and before Sybase was concerned about crafting definitions to ensnare Vertica. The guide confirms that page chains use references in the pages to point to the next page in the chain. The guide also confirms that a collection of pages accessed only via an index is **not** a page chain. *See Phillips v. AWH Corp.*, 415 F.3d 1303, 1322 (Fed. Cir. 2005) (en banc) (describing proper use of contemporaneous extrinsic evidence, especially evidence from an "unbiased source 'accessible to the public in advance of litigation'") (quoting *Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1585 (Fed. Cir. 1996)).

IV. THE PROPER CONSTRUCTION OF THE "PAGE" TERMS

Three terms are particularly significant and go to the heart of the alleged invention claimed in the '229 patent. These three terms will likely require the most time at the claim construction hearing. These terms are "Data Page," "Linking Together," and "Page Chain."

A. "Data Page"

Construction: A fixed unit of physical storage.

1. Support For Vertica's Claim Construction

Central to every claim of the '229 patent is the use of "data pages" as the medium to store data on disk or similar mass storage device. Exemplary claim 1 is recited here for reference:

> 1. In a computer system having a database storing a database table, said database table presenting user information as rows of data records divided into columns of information, each column representing a particular category of information, a method for storing the database table by vertically partitioning each and every column of the database table without regard to storage devices available to the system, the method comprising:
>
> irrespective of storage devices available to the system, creating for each column of the database table at least one associated **data page** for storing data values for the column;
>
> for each data record of the database table, storing the data value for each column of the data record in one of the at least one **data page** associated with the column, so that data values for each particular column of a database table are all stored together in at least one **data page** associated with said each particular column; and
>
> for each column, linking together all of said at least one **data page** associated with the column to form a page chain for the column, so that each page chain stores together all of the data values for a particular column of the database table.

This claim requires the data of a database to be stored in one or more "data pages."

The term "data page" was a common technical term in the database art well before the alleged invention of the '229 patent. It thus has an **ordinary meaning** that exists apart from the patent. A data page is a fixed unit of physical storage. No more and no less. No special or unique definition of "data page" is provided in the '229 patent. As the patent acknowledges many times, a "data page" is not unique to the '229 patent—the prior art also used data pages. Accordingly, any definition of "data page" **must** read on the prior art data pages disclosed in the '229 patent. Sybase's construction must be rejected because it attempts to redefine the term to specifically **exclude** prior art data pages.

The prior art's use of data pages is illustrated in Figure 3A from the '229 patent.

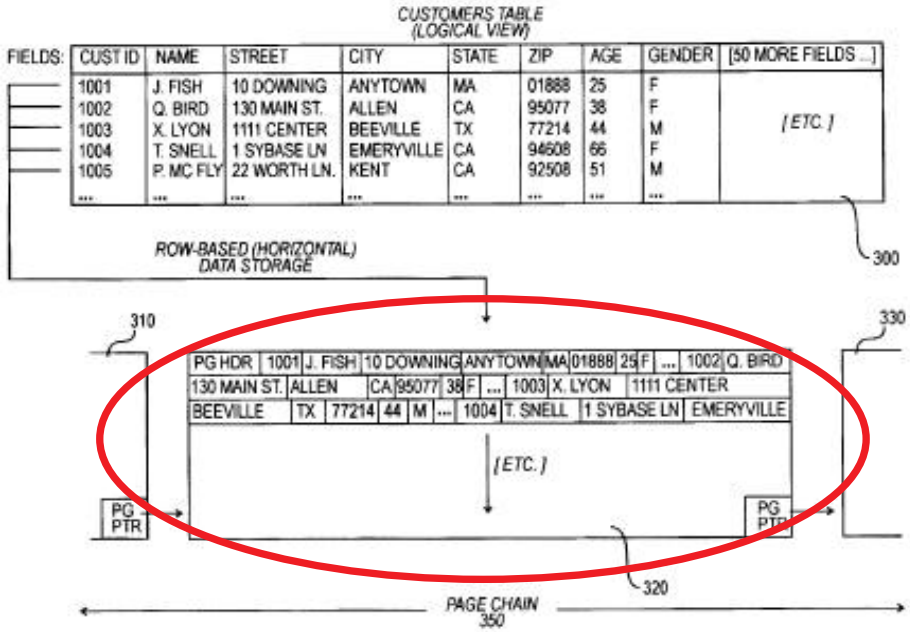


FIG. 3A

The patent explains: "as shown in FIG. 3A, row-based (horizontal) data storage of the table 300 entails storing each row in a **data page, such as data page 320**." (Ex. A, '229 patent, col. 12, ll. 62-64) (emphasis added).

According to the '229 patent, the difference between the alleged invention and the prior art is that the old-fashioned data pages contain rows of data instead of columns of data: "Instead of storing the data members of each row together on a **data page**, the data members for each column are stored together as 'cells' on a **data page**." (*Id.* at col. 13, ll. 13-15).

Because the '229 patent does not use the term "data page" in a special way, it is the role of the Court to determine what the ordinary technical meaning of the term is. The inquiry begins with the term itself. Outside the computer field, there is a lay understanding of a "page"—which is a leaf in a book. A book is formed by binding together pages having a fixed and equal size. The page is something physical. It can be thought of as a small container for a subset of the words contained in the whole book. A page has a similar connotation in the database field.

Instead of a book, the storage medium is typically a disk.[10] Like a book, the disk (or other storage device) can be divided into pages.

The "data page" is a unit of physical storage. It is a physical area on a disk or other storage medium: "the actual **physical** database itself" is "the records contained in **data pages** on a storage device." (*Id.* at col. 1, ll. 38-39). This is confirmed in the discussion of the Buffer Manager, where the patent explains that data is stored at a specific "physical address or page number" on the disk:

> [W]hen the Create method is called, the system does not at that point find a **physical location on disk to store the object**. Only upon occurrence of an explicit write call (e.g., when "paging out") does the system invoke the page allocator for putting the information in one of the files which the object serves as the Buffer Manager for. . . . When written to disk, if the object has not been allocated **a slot on disk** (i.e., **a physical address or page number on disk**), the Manager will (in conjunction with a page manager) allocate one.

(*Id.* at col. 17, l. 63 to col. 18, l. 19).

The '229 patent specifically distinguishes a "logical" table (which is conceptual only) from the "chain of pages (i.e., **physical** chain of pages) which represents the table." (*Id.* at col. 14, ll. 55-64 (emphasis added)). For example, when discussing a "B-Tree Index," the patent distinguishes between the "key," which is a "logical" entity, from the "physical page number" into which the key is translated. (*Id.* at col. 15, ll. 27-36). The claims use "data page" because a physical structure is claimed–not a logical structure.

The figures in the '229 patent support this conclusion. Figure 3A, for example, which purports to disclose a prior art system much like that found in the Date textbook distinguishes

[10] The patent uses a disk as the example of a storage device throughout the entire patent, although at one point, the patent mentions "flash memory" as an alternative to a disk. (Ex. A, '229 patent, col. 5, ll. 29-36).

between the "logical view" on the top of the figure and the actual physical storage on disk shown in the bottom half of the figure:

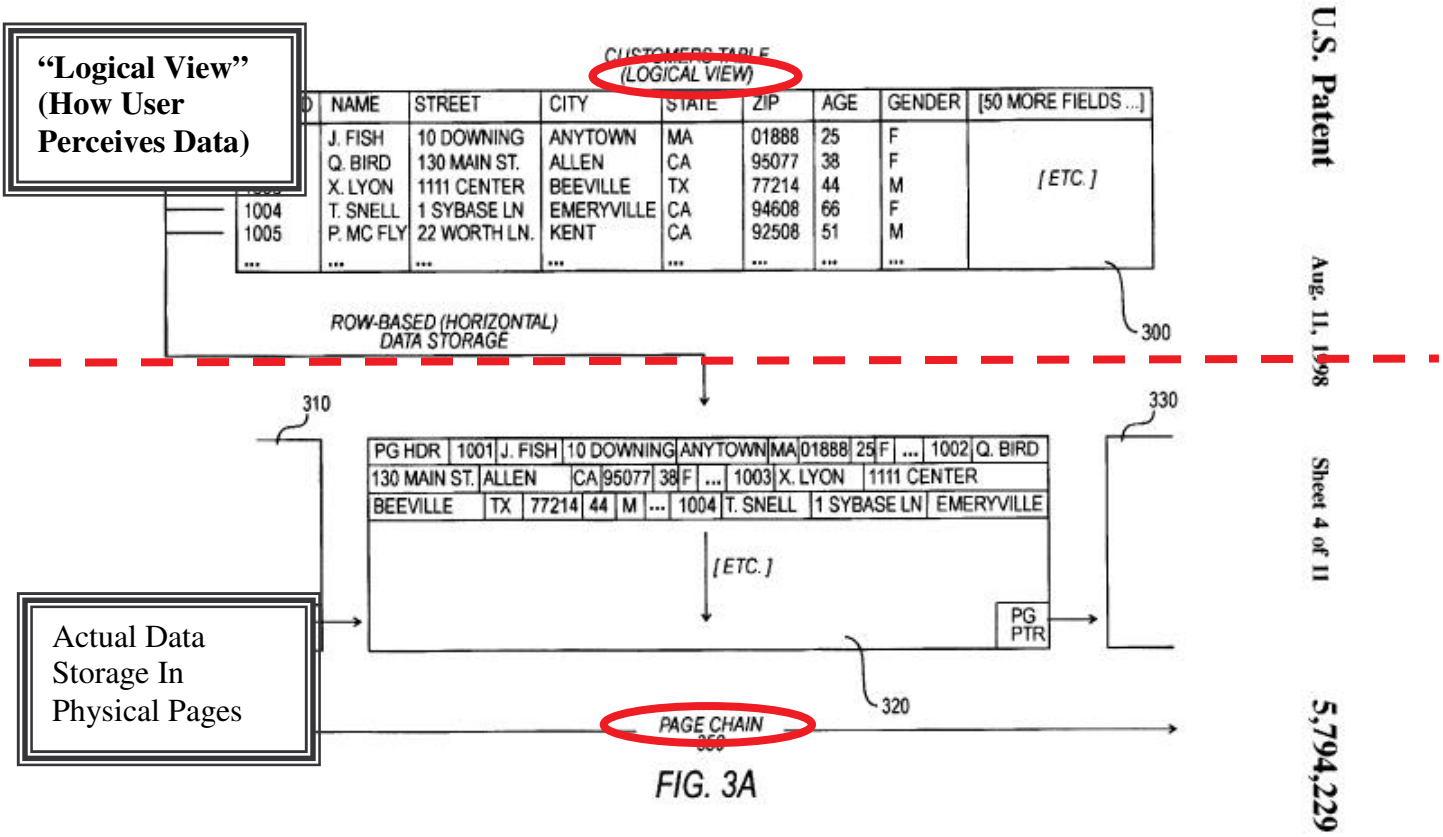


FIG. 3A

Finally, when discussing large block transfers, the '229 patent states that "data stored on the **now larger page** [is] stored contiguously on a storage device (disk). As a result, retrieval is optimized by transfer of a **single contiguous unit** (e.g., disk sector)." (Ex. A, '229 patent, col. 10, ll. 18-22) (emphasis added).

The Date textbook, cited by the '229 patent as a general text on databases, confirms that the ordinary meaning of the term "data page" is a fixed unit of physical storage."[11] The Date

[11] The C.J. Date textbook is intrinsic evidence for purposes of claim construction because its contents were specifically cited in the patent. (Ex. A, '229 patent, col. 1, ll. 51-55). In particular, Sybase included the Date textbook to explain "[t]he general construction and operation of a database management system." (Id.) By citing this book, it becomes part of

textbook explains that a disk manager "determines the **physical** location of the desired page on the disk." (Ex. J, C. J. Date, An Introduction to Database Systems, Volume I, Fifth Edition (Addison-Wesley 1990) (the "Date textbook"), p. 57 (emphasis added)). The Date textbook also draws a distinction between "physical" entities and "logical" ones. For example, the Date textbook describes "logical" "page sets" which are conceptual groupings of pages. The page sets are "logical" because they can include pages from all over the disk—the "page sets" have no relation to the physical location of the pages. The Date textbook distinguishes the logical "page sets" from the physical pages themselves: "the disk manager needs to know exactly where page *p* is on the physical disk" and "[t]he mapping between page numbers and physical disk addresses is understood and maintained by the disk manager." (*Id.* at pp. 58-59).

Further support is found in a prior art article which confirm that the construction "a fixed unit of physical storage" captures the **ordinary meaning** of the term data page. *Callaway Golf Co. v. Acushnet Co.*, 576 F.3d 1331, 1338 (Fed. Cir. 2009) ("evidence of accepted practice within the art, when not at variance with the intrinsic evidence, is relevant to the question of how a person of skill in the pertinent field would understand a term.").

For example, a reference by Hammer and Niamir entitled "A Heuristic Approach to Attribute Partitioning," published in 1979, discusses "well-known" database techniques. It states that "attribute partitioning" (i.e., vertical partitioning) of a database was a "well-known database technique that seeks to improve database system performance by reducing the amount of unneeded information that is brought into main memory in block transfers from secondary storage." (Ex. K, Michael Hammer et al., "A Heuristic Approach to Attribute Partitioning," Proceedings of the 1979 ACM SIGMOD international conference on Management of data, May

---

the intrinsic record that the Court should consider in construing the patent. *See Cook Biotech Inc. v. Acell, Inc.*, 460 F.3d 1365, 1376 (Fed. Cir. 2006).

30 - June 01, 1979, Boston, Massachusetts, p. 93). This article–written more than a decade before the '229 patent–distinguishes the "logical" table from the physical (that is, actual) data storage. It states, "[t]he concept of attribute partitioning is to break a logical file of logical records into several physical records stored in different physical subfiles, each containing only a subset of the attributes (fields) of the original record." (*Id.*) Each file "is assumed to reside on a direct access storage device like a disk, which is divided into **fixed size blocks called pages**." (*Id.* at 95 (emphasis added)). The article also explains that a "page is the unit of transfer between secondary and primary memory; the size of a page (in words) is fixed as a system parameter and the cost of accessing a page is constant." (*Id.*)

The Date Textbook further supports this understanding, when it states that each page set consists "of a collection of **fixed-size pages**." (Ex. J, Date textbook, p. 58).

The '229 patent also discloses the use of fixed size pages. For example, in the "preferred embodiment" of the '229 patent, each data page in the system is 64 kilobytes. (Ex. A, '229 patent, col. 17, ll. 54-61). Sybase suggests that the '229 patent discusses variable-size pages, but the citations do not support Sybase's position. For example, Sybase cites to a reference to a 4k block. This is not a 4k "page." The patent states that 16 4k blocks are stored together on a 64k page. (Ex. A, '229 patent, col. 17, ll. 57-62).

Thus, the '229 patent, the Date Textbook cited by the '229 patent as an authoritative text on the database management systems, and the prior art draw a clear distinction between "logical" or conceptual entities and physical entities, and data pages are universally physical areas on a disk or other storage device. They are of a fixed size for any given system. They are not, as Sybase argues, merely a conceptual or "logical" concept. Because the patentee did not "act as its

own lexicographer," the ordinary meaning of "data page" in the art must control. *Helmsderfer v. Bobrick Washroom Equipment, Inc.*, 527 F.3d 1379, 1383-84 (Fed. Cir. 2008).

2. Critique of Sybase's Construction

Sybase's proposed construction of "data page" as a "logical structure for containing one or more cells (i.e., column value for a record)" is incorrect because the '229 patent expressly says that a data page is a "physical" unit of storage—not a "logical structure." As explained above, a "data page" is something real—it is something physical. It is where the actual data resides on the storage device. A data page is specifically distinguished from a "logical" or conceptual structure such as a "table" representation.

The distinction between "logical" structures and "physical" pages is apparent from Figures 3A through 3C of the '229 patent. In Figure 3B, for example, the "logical" structure is at the top and the physical (that is, actual) structure is at the bottom. (Ex. A, '229 patent, col. 12, l. 51 to col. 13, l. 3; col. 13, ll. 12-25).

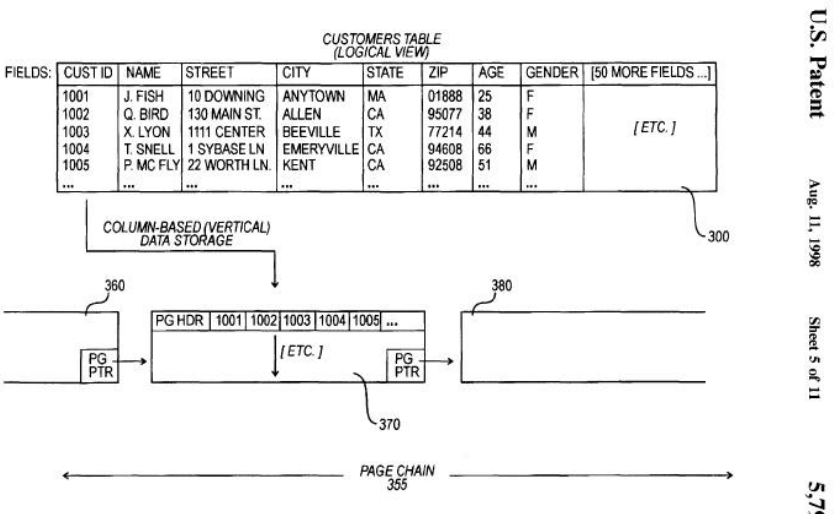


FIG. 3B

Sybase's proposed construction is incorrect because it focuses on the top half of the figure (which contains the logical structure) and ignores that the "data pages" are found in the

lower half of the figure (which describes the physical structure). Likewise, Sybase cites to two sentences from the specification as alleged support for its "logical structure" terminology. However, both citations merely state that a "table" is a "logical" concept—a point that is undisputed. The "data pages" in the prior art and in the '229 patent are the **physical** storage units on the storage device that **hold** the "logical" table – the logical table is not the data page.[12]

Sybase's inclusion of the term "cell" in its construction—"one or more cells (i.e. column value for a record)"—is incorrect for at least five independent reasons.

First, the '229 patent uses the term "cell" in two mutually exclusive ways to mean either (a) a piece of column data, (*see, e.g.,* the abstract*)* and (b) an area in *some* data pages that holds a piece of column data. (*see, e.g.*, col. 13, ll. 29-31). Because the term "cell" is ambiguous, it should not be used in a claim construction. *See O2 Micro Intern. Ltd. v. Beyond Innovation Technology Co., Ltd.*, 521 F.3d 1351, 1362 (Fed. Cir. 2008) (claim construction should add clarity and enhance understanding of the claim terms).

Second, the inclusion of the term "cell" would appear to **exclude** prior art data pages. Such exclusion is incorrect. The patent describes prior art data page 320 as containing a "collection of particular data values for that row." (Ex. A, '229 patent, col. 13, ll. 1-3). Data pages can store both row-based data and column-based data and the inclusion of the term "cell" (which Sybase equates with column-based data) would be improper.

Third, if the word "cell" means "column value for a record," it would render other claim terms superfluous. Claim 1 states "at least one associated data page for storing data values for the column." If a "data page" inherently stored "cells (i.e., column value for a record)" there

[12] Moreover, the term "logical structure" appears vacuous. What exactly is a "logical structure"? What would not be a logical structure? If anything and everything is a "logical structure," the claim term would be superfluous.

would be no need to add "for storing data values for the column" to the claim. Moreover, the inclusion of "i.e." in a claim construction is rarely appropriate and should raise a red flag—why include two definitions instead of just one?

Fourth, nothing in the independent claims requires that the page store "cells." When the patentee intended to limit a claim to "cells," it did so explicitly. Claim 19 for example, states "wherein the data pages . . . store data values in fixed-length cells." Claim 1 does not require "cells." Claim 16 does not require "cells."

Finally, Sybase's proposed construction is wrong because it would exclude certain embodiments claimed in dependent claims. Sybase states that the term "cell" refers to the "**actual** data values" and not mere representations of data values. (Dkt. No. 107, Sybase's Claim Construction Brief, pp. 17, 20). Accordingly, a "cell" must be the "actual" data value and not a compressed value that must be decompressed to recreate the original data. However, this position cannot be squared with dependent claim 15 which requires data pages to store compressed data (and not "cells" as Sybase interprets that term).

For example, the "LZW" compression patent disclosed and incorporated into the '229 patent states: "Data compression refers to any process that converts data in a given format into an **alternative format** having fewer bits than the original." (Ex. L, U.S. Patent No. 4,558,302, col. 1, ll. 14-16). The "LZW" compression algorithm uses a table lookup substitution scheme that stores "digital code signals" in place of the "digital data signals." (*Id.* at col. 4, ll. 54-60). A "digital code signal" would not be a "cell" as Sybase interprets that term.

Accordingly, a data page is a unit of physical storage and not merely a "logical structure." Data pages do not require "cells," a term that is both ambiguous and confusing. The

Court should adopt Vertica's construction because it best captures the ordinary meaning of that term, is faithful to the patent, and avoids the errors of Sybase's proposal.

B. "Linking Together" And "Page Chain"

"Linking Together"
Construction: *Connecting a series of pages one after another.*

"Page Chain"
Construction: *A sequence of pages connected by a reference each preceding page to each successive page*[13]

1. Support For Vertica's Claim Construction

In claim 1, the terms "linking together" and "page chain" are sufficiently intertwined that it makes sense to discuss the terms together. The "linking together" in claim 1 produces a specific result—it produces a "page chain."

The relevant phrase from claim 1 is: "**linking together** all of said at least one data page associated with the column **to form a page chain** for the column."

"Linking together" each element of a chain involves connecting each element to its neighboring elements. Once all connections are in place, the result is a "chain." The act of "linking together" the data pages is illustrated in Figures 3A-C of the '229 patent. Each figure illustrates a pointer in one page that points to the next page in the series. For example Figure 3A illustrates a page pointer ("PG PTR") linking page 310 to page 320:

[13] This definition is a slight modification to the originally proposed definition, which had a flaw. It is incorrect to say that "each" page must include a reference to the successive page because the last page has no successive page. Therefore, correct this error, "each page" has been changed to "each preceding page."

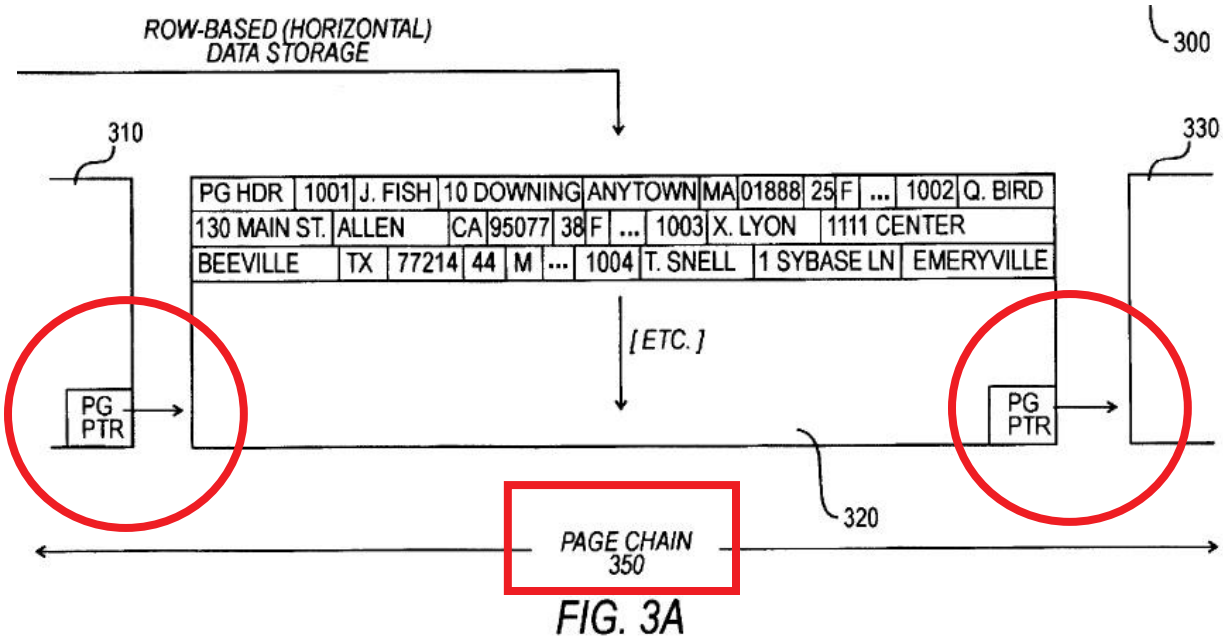


FIG. 3A

In the '229 patent, the only disclosure of how page chains are formed by linking together pages are Figures 3A-C and their accompanying descriptive text. Page chains are uniformly described as a series of pages with each page containing a link connecting it to a neighboring page.

Linking pages to form page chains existed in the prior art, and the '229 patent describes prior art page "linking." The patent describes what was "conventionally" done in the art. By "conventionally," the patent means "ordinarily." The use of the adjective "conventional" with linking provides a strong indication of what the patentee understood the term "linking" to mean in the database field.

The '229 patent explains that in the prior art each page is connected to its neighboring page. Because the pages are in sequence, the depicted pages have a "left neighbor" and a "right neighbor." The patent states that the "actual connection" is "typically achieved using conventional page pointers, such as forward-referencing page pointers." (Ex. A, '229 patent, col. 13, ll. 4-11).

The data page **320** is conventionally connected to other data pages, such as data page **310** (left neighbor) and data page **330** (right neighbor). In this manner, the pages form a single page chain **350**. Actual connection between the pages is typically achieved using conventional page pointers, such as the forward-referencing page pointers shown for page **310** (i.e., which points to page **320**) and for page **320** (i.e., which points to page **330**).

The patent does not distinguish the claimed linking from the prior art; nor does it offer a special meaning of "linking together" data pages to form a "page chain." On the contrary, the '229 patent states that the invention uses the same "linking" that was used in the prior art. The '229 patent states that the data pages "may be connected or linked to other data pages (in a manner similar to that previously shown)." (Ex. A, '229 patent, col. 13, ll. 32-35).

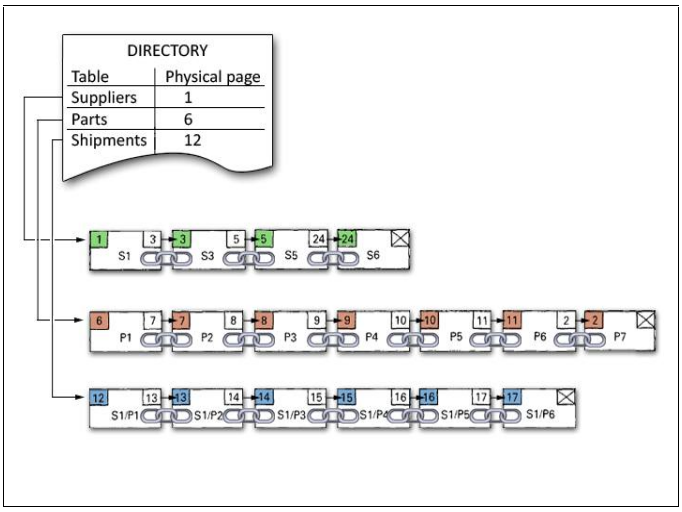
The patent discloses precisely how a "page chain" is formed: "The data page 320 is conventionally connected to other data pages, such as data page 310 (left neighbor) and data page 330 (right neighbor). **In this manner, the pages for a single page chain** 350." (Ex. A, '229 patent, col. 13, ll. 4-7 (emphasis added)). The '229 patent does not state "this is one way of forming a page chain." The patent states that this is how page chains are formed. The '229 patent does not disclose any other method for forming "page chains."

Vertica is not reading in the "preferred embodiment" of the invention, as Sybase argues. Vertica is properly interpreting the term "page chain" to reflect the uniform meaning of that term throughout the patent, including descriptions of the prior art, while also taking into account the prosecution history. The '229 patent's reference to what was "conventionally" done in the prior art is strong evidence of the meaning of these terms, especially where the patent does not distinguish the claimed "linking together" from the prior art.

The '229 patent uniformly discusses forming "page chains" in a single way, whether those "page chains" were in the prior art, in the "preferred embodiment," or in any other

embodiment of the invention. A reference (typically a pointer) is placed in a data page to link or connect each data page to each neighboring page. All data pages are "linked together" by connecting (via the page references) each page to each neighboring page in a series, one after another. The uniform disclosure of the '229 patent strongly supports Vertica's proposed claim constructions. The '229 patent contains no discussion of other ways of "linking together" data pages to form a "page chain."

Sybase's choice of the term "chain" to describe the "linking together" also supports Vertica's construction. The term "chain" connotes the linking of objects in series—one after another. Vertica's Technology Tutorial used the following illustrative graphic:



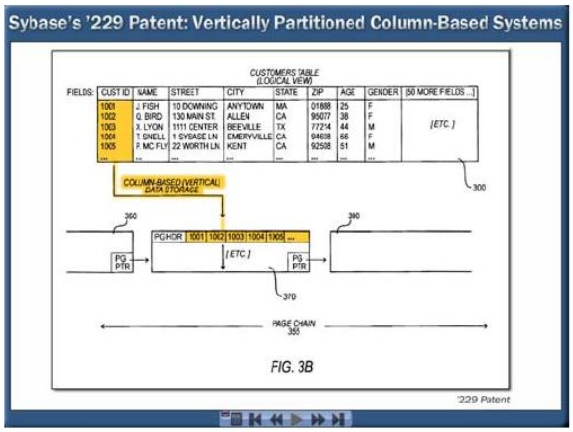
As shown, links (called page pointers in the '229 patent) are inserted into the data pages that connect each page to neighboring pages. The result is a sequence of pages connected by a reference in each preceding page to each successive page. When these sort of links are used, the pages truly do form a "chain" as illustrated in the Tutorial.

A "chain" is much more specific and narrow than a "collection." Sybase's vague construction (a "collection" of associated data pages) fails to give proper weight to the choice of the word "chain" in each claim of the '229 patent. Assume someone separated each ring from

two standard bicycle chains and put all the metal rings from the first chain in a first bowl and all the metal rings from a second chain in a second bowl. Each "bowl of rings" would be a "collection" of rings. Each ring in each bowl is clearly "associated" with the other rings in the bowl by virtue of being in the same bowl. However, each bowl of rings is not a "chain." The "collection" of rings would only become a chain once again if each ring were connected to a neighboring ring to form a series of connected rings.

Sybase's own Technology Tutorial supports Vertica's construction because it also states that "a series" of pages is connected to form the page chain:

Figure 3B illustrates this form of page storage. As can be seen in figure 3B, the data values from the first column of the Customers Table are stored in a data page. A series of pages containing just these data values from this column forms a page chain.



Far from being a "preferred embodiment," the act of "linking together" is uniformly shown as connecting a series of data pages one after another via **references in the pages themselves**.

The prosecution history also supports Vertica's construction. As discussed in the Background section of this brief, the Examiner rejected the claims by combining two references. The Examiner stated that the Naecker reference failed to disclose the "linking" of data pages. (Ex. F, Office Action dated July 18, 1997, p. 6, ¶ 14). The Examiner said this even though the data for each column was clearly "associated" in some manner. Specifically, the Naecker reference disclosed a "system catalog" for keeping track of the data in the columns. The

Naecker reference also disclosed a B-Tree index. But, the Examiner found that the B-Tree index, which clearly "associated" the data pages, did not disclose the "linking" aspect of the claim. According to the Examiner, "linking together" data pages is more precise than merely "associating" a "collection" of data pages (via an index, for example).

If "linking together" merely meant "associating" a "collection" of data pages, then the Patent Office would have found the '229 claims anticipated by Naecker. Because the Examiner interpreted "linking" as being more specific than "associating," the Examiner cited a further reference for the "linking" aspect of the claim. For this "linking" aspect, the Examiner relied on Snellen, which disclosed the use of page pointers to point from one page to an overflow page. (*Id.* at p. 6, ¶ 16). Thus, the prosecution history supports Vertica's construction: "linking" is not so broad as to encompass any "association."

Extrinsic evidence confirms the correctness of Vertica's claim construction. When asked to explain what a "linked page" meant in the database field (not what it meant specifically in the patent), Mr. Peter White, one of two named inventors on the '229 patent, provided the following testimony:

> Q. What's a linked page?
>
> A. The typical form of a linked page is you have information in the page about where the next page in the chain is, and so they're linked together in that form.
>
> Q. I guess you're saying the downside of using a linked page chain is that you need to go from the first page and step through each subsequent page to ultimately get to the page you're looking for?
>
> A. Yes.

(Ex. I, Deposition of Peter White, p. 42, ll. 1-10).

Mr. White specifically distinguished linked pages from a collection of pages associated by an index such as a B-Tree or similar tree index. (*Id.* at p. 40, l. 24 to p. 42, l. 10). Thus, B-Tree indexes do **not** create page chains. To confirm, the questioner asked the following follow-up question about the use of indexes resembling the B-Tree:

> Q: Now, you mentioned that this [a tree index] is an alternative to linked pages, right?
>
> A: Yes.

(*Id.* at p. 41, ll. 22-24).

Mr. White specifically mentioned that one drawback of page chains was that they could be slow because it was necessary to hop from one data page to the next and to the next to find the data page you were interested in. Such testimony confirms Vertica's understanding of "page chains" which is a series of pages in which each preceding page **includes a reference** pointing to the next page in the chain.

> Q. And you're saying that because with a page chain you start at the first page, go to the next page, and go to the next page in sequence?
>
> A. Yes.
>
> Q. And so if you want to get to the last page, you're going to have to then jump through each page?
>
> A. Yes.
>
> Q. But if you only want -- If the page you're looking for is one-third the way through the bitmap, then you only have to jump through the first third of the pages?
>
> A. Yes.
>
> Q. And so that makes page chaining slow?
>
> A. Yes.
>
> Q. Why did you use it for the FASTPROJECTION index?

> A. By the time we would be using the FASTPROJECTION index, we knew exactly which rows we were interested in and we had them in row order, so it made a lot more sense to go through a page chain to get at the data.

(*Id.* at p. 202, l. 5 to p. 203, l. 1). Mr. White's testimony confirms the conventional definition of page chains as construed by Vertica, discussed in the '229 patent, contained in books, the prior art, and, as will now be explained, in Sybase's own literature.

Since at least 1999, Sybase has published its "Adaptive Server Performance and Tuning Guide," which is presently available for download on Sybase's website. The manual states that pages are "linked" by storing **pointers in the data pages** pointing to neighboring pages. The Sybase system being discussed uses a "doubly-linked list"[14] meaning that it uses both "forward-referencing pointers" as disclosed in the '229 patent and also "backward-referencing pointers."

Lock Schemes and Differences Between Heaps

> The data pages in an allpages-locked table are linked into a doubly-linked list of pages by pointers on each page. Pages in data-only locked tables are not linked into a page chain.
>
> In an allpages-locked table, each page stores a pointer to the next page in the chain and to the previous page in the chain. When new pages need to be inserted, the pointers on the two adjacent pages change to point to the new page. When Adaptive Server scans an allpages-locked table, it reads the pages in order, following these page pointers.

[14] Vertica's Technology Tutorial explained that a "list" in the software arts is a data structure having elements with two parts: data and a pointer to the next element of the list. The pointers link the elements thereby imposing a sequence, and hence, a "list."

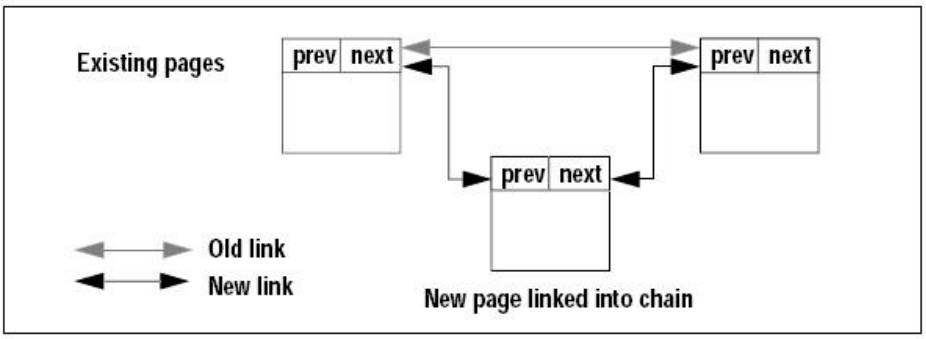


Figure 3-5: Page linkage in allpages-locked tables

(Ex. D, Sybase Adaptive Server Performance And Tuning Guide, pp. 3-11 to 3-12). Pages linked in the page chain are scanned by jumping from page to page following the page pointer to the "end of page chain":

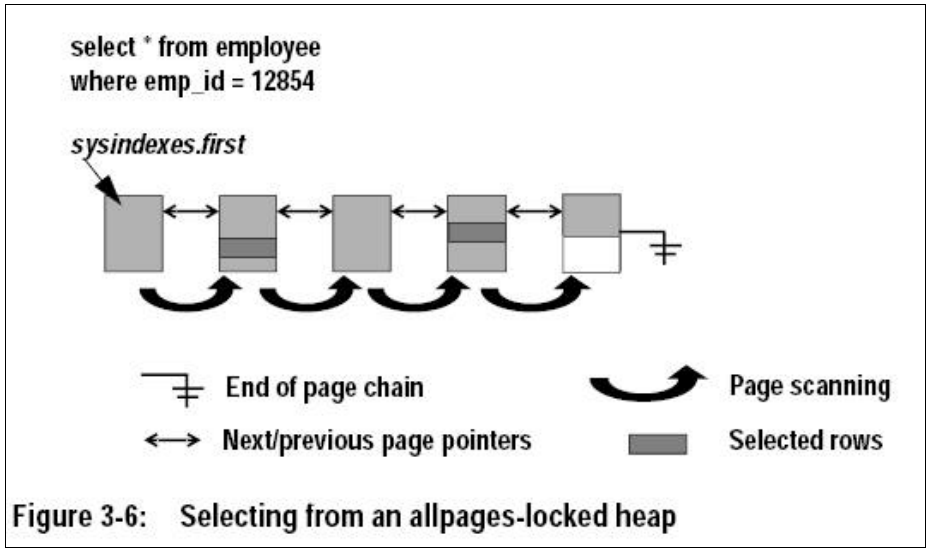


Figure 3-6: Selecting from an allpages-locked heap

(*Id.* at 3-13).

Sybase's manual specifically distinguishes these **linked** pages from a **collection** of pages that are merely **associated** using an index (such as an allocation page). In fact, the manual states that the collection of pages associated by an index **are not linked into a page chain**:

> Since the pages of data-only-locked tables are not linked in a page chain, a **select** query on a heap table uses the table's OAM and the

allocation pages to locate all the rows in the table. The OAM page points to the allocation pages, which point to the extents and pages for the table.

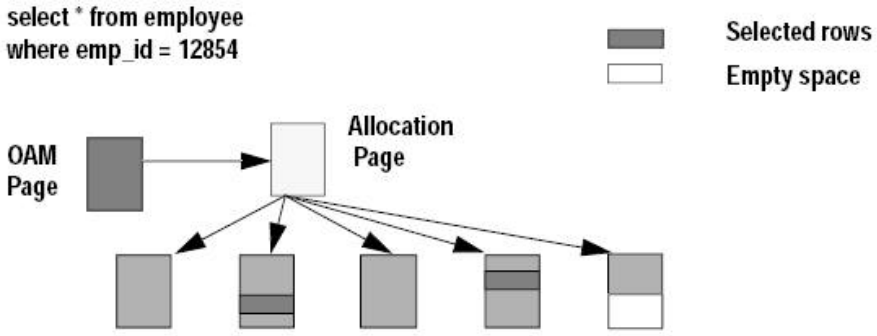


Figure 3-7: Selecting rows from a data-only-locked table

(*Id.* at pp. 3-11 to 3-12). Sybase's own manual thus undermines Sybase's overly broad claim construction in which any "collection" of "associated" pages are "page chains." Under Sybase's construction, anything would be a "page chain" and thus the term "chain" would be read out of the claim.

2. Critique of Sybase's Construction

Sybase's constructions are incorrect because they seek to rewrite precise terms to cover something other than what Sybase invented and claimed. Sybase argues that the term "linking" means "connecting or associating" and that a "page chain" is "a collection of one or more associated data pages." In both instances, Sybase has substituted general and vague language for the specific terms used by those skilled in the database field. The Court should reject Sybase's constructions because they fail to capture the ordinary meaning of these terms.

For example, "linking" does not mean just any "connecting" and does not mean "associating." In fact, the inclusion of ". . . or associating" in Sybase's "linking" construction is improper for at least three reasons.

First, Sybase's use of "associating" and "connecting" together creates ambiguity because "connecting" is narrower than "associating," and "linking" can not be equated to both terms. Whereas the terms "linking" and "connecting" both connote some sort of attachment between the data pages, the term "associating" connotes a mere relationship. In the context of a database, "associating" is too vague to be meaningful. All data in a database is "associated" in some way. "Associating" should not be included in the construction.

Second, use of the word "connecting" in isolation invites the same (or at least similar) mischief as "associating." Sybase is sure to argue that B-Tree indexes and catalogs "connect" pages by "associating" them. Such an argument is wrong, and Vertica's construction aims at precluding it. As discussed above, any argument that B-Tree indexes and catalogs are covered by the claims is inconsistent with the prosecution history, and is contrary to the '229 patent disclosure and usage in the art.

Third, when Sybase wished to say "associating" it did so explicitly. In claim 1, for example, the "linking" step itself uses the term "associated:" "**linking together** all of said at least one data page **associated** with the column." Had Sybase meant to say "**associating** all of said at least one data page associated with the column," it would clearly have said so explicitly—as demonstrated by the use of the term "associated" in the very clause at issue. It didn't. Also, why would a step of "associating" be necessary if the pages were already "associated with the column?" It wouldn't. Sybase chose the much narrower term "linking together." The conscious choice of different terms is presumed to convey different meanings. *Board of Regents of the University of Texas System v. BENQ America Corp.*, 533 F.3d 1362, 1371 (Fed. Cir. 2008) ("Different claim terms are presumed to have different meanings").

Finally, Sybase implies (but does not outright say) that something called the "BArray" connects data pages. It does not. As Sybase accurately quotes, the "BArray" "presents a view of a contiguous array of database pages." (Ex. A, '229 patent, col. 45, ll. 35-38 (emphasis added)). Thus it is simply a "view"—a "logical" way to represent, in memory, the pages that exist on the disk—and presents them "logically" as a contiguous set (even though they are not physically stored that way). The "Barray" are only disclosed as a data member of the "hs_dp" column object, which "relies on its Buffer Manager for performing the necessary I/O operations for retrieving data pages from the disk subsystem." (Ex. A, '229 patent, col. 43, ll. 50-53). Therefore, the Barray is at most a copy of data fetched by the Buffer Manager. It cannot access the disk directly and therefore cannot "link" pages together to form a "page chain." Morevoer, nothing in the '229 patent says that the BArray stores data in "page chains" or that the "BArray" performs the function of "linking together" data pages or forms "page chains." The single sentence cited by Sybase which neither mentions "linking" or "page chains" cannot help Sybase.

3. A Few Words Regarding Sybase's Criticism Of Vertica's Constructions

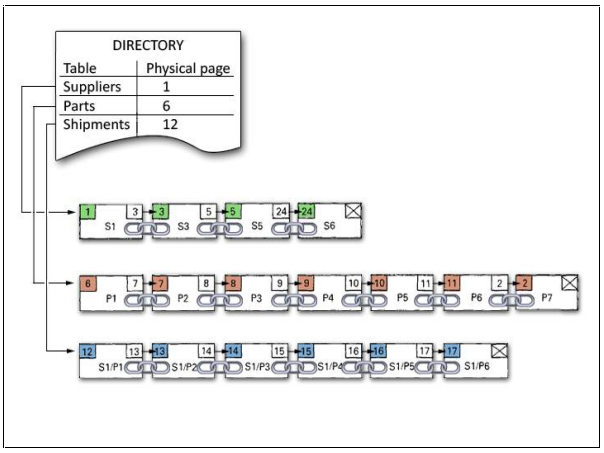
The doctrine of claim differentiation does not apply. Sybase argues (incorrectly) that Vertica's construction is "taken from a dependent claim." It is not. There is a dependent claim (claim 6) that refers to "page pointers" in "each" data page. Vertica's construction does not require "page pointers" per se. Vertica's construction also does not require a "page pointer," or even some other type of link, in "each date page." For example, if only forward-referencing pointers are used, there would be no pointer in the last page. It has nothing to point to. Thus, claim 6 appears limited to the special case where forward-referencing and backward-referencing pointers are used. In such a narrow circumstance, there would be a pointer in "each data page." Accordingly, claim differentiation does not apply as Vertica's proposed construction does not

render claims 1 and 6 of equal scope. *Sinorgchem Co., Shandong v. International Trade Com'n*, 511 F.3d 1132, 1140 (Fed. Cir. 2007).

Vertica does not construe these terms to require "page pointers." Sybase criticizes Vertica for limiting the "linking" to "page pointers." However, this criticism is misplaced, as the term "page pointers" is not in Vertica's construction.

Sybase also asserts that Vertica "excludes" embodiments because the '229 patent mentions a "system catalog," a "B-Tree," and a "block map" as alternate techniques for "associating" pages. These embodiments are not "excluded." Each of these "alternatives" work both with systems using pages chains and those that don't. None actually form page chains by linking pages together.

A "system catalog" is used to locate the first page of a page chain, when page chains are used:



Nothing in the patent suggests that a "system catalog" forms a "page chain."

A "B-Tree" index may be used to quickly jump to the middle of a group of pages, regardless of whether those pages are stored as a chain. The B-Tree index is **in addition** to the references stored within each page pointing to the successive page in the chain. (Ex. A, '229 patent, col. 15, ll. 28-36) (stating that the random access provided by B-Trees is "in addition to

sequentially scanning the pages" provided by the page pointers). B-Trees are never described in the patent as forming "page chains." Moreover, the testimony of Peter White distinguished B-Trees and page chains.

The "block map" is used only in systems that have compression and the "block array" presents a "view" into the data (much like the "logical table"). Both may be used equally with pages stored as page chains as those stored in another manner. Neither "block maps" nor "block arrays" link together pages to form page chains, and nothing in the '229 patent suggests that they do.

The '229 patent uniformly correlates the "linking together" of pages to form a "page chain" to the forming of a sequence of pages by placing a reference (such as a pointer) in each preceding page to each successive page.

V. THE REMAINING TERMS

A. "Computer System" / "Database System"

Construction: No construction necessary, or in the alternative, "a system including a computer" and "a system including a database."

1. Support For Vertica's Claim Construction

The claim language is clear—the invention is database software running on a computer system. The jurors will know what a computer system is and no further construction is necessary. *Phillips v. AWH Corp.*, 415 F.3d 1303, 1314 (Fed. Cir. 2005) ("In some cases, the ordinary meaning of claim language as understood by a person of skill in the art may be readily apparent even to lay judges, and claim construction in such cases involves little more than the application of the widely accepted meaning of commonly understood words."). The terms "computer system" and "database system" are significantly more clear that the constructions proposed by Sybase.

2. Critique of Sybase's Construction

Sybase's construction is incorrect because the term "computer system" does not mean a "relational database management system." A relational database management system is a type of data processing system, which is a type of database system, which is a "common mechanism for storing information on computer systems." (Ex. A, '229 patent, col. 1, ll. 23-31). It is clear that the terms "computer system" and "relational database management system" are not coextensive. Moreover, Sybase's construction will only create confusion, as it will then require the Court to determine what is meant by "relational database." The very fact that Sybase felt compelled to include dictionary definitions for terms within its proffered construction (specifically, for the term "relational") demonstrates the impropriety of accepting Sybase's construction. There is no need to construe the terms "computer system" and "database system."

B. "Without Regard To Storage Devices Available To The System"/ "Irrespective Of Storage Devices Available To The System"

Construction: Storing each column without taking into account the configuration of the storage devices.

1. Support For Vertica's Claim Construction

The plain meaning of "without regard to the storage devices" is that the database has no control over how the columns are actually divided among the available storage devices. The size, speed, and other aspects of the storage device are ignored. That is what it means to do something "without regard to the storage devices."

The parties agree that this claim language was added by Sybase during prosecution to overcome a rejection by the Patent Office. (Dkt. No. 107, Sybase's Claim Construction Brief, p. 14; Ex. H, Response to Office Action dated January 16, 1998, pp. 9-10). In response to the Patent Office, Sybase amended its claims and made two statements explaining why its application was allowable, both of which are relevant to claim construction. First, Sybase said

that its invention achieved superior performance "without placing a particular column on a particular disk (and another column on another disk, and so forth and so on)." (Ex. H, Response to Office Action dated January 16, 1998, p. 8). Second, Sybase argued that "the present invention is to break-up the data up vertically regardless of the disk configuration. . . . Quite simply, the disk configuration is not important to the operation of the present invention." (*Id.* (emphasis in original)).

In construing this claim element, the Court must take into account **both** of Sybase's statements. *Andersen Corp. v. Fiber Composites, LLC*, 474 F.3d 1361, 1374 (Fed. Cir. 2007) ("An applicant's invocation of multiple grounds for distinguishing a prior art reference does not immunize each of them from being used to construe the claim language"). Sybase's second argument—ignored in Sybase's Claim Construction Brief—also limits the claim and the argument is clear that the invention stores the data "regardless of the disk configuration."

Sybase seeks a narrow construction based only on its first statement; but Sybase may not ignore one of the two arguments it made to distinguish Naecker and Snellen—both statements must be taken into consideration. *See Id.*; *Norian Corp. v. Stryker Corp.*, 432 F.3d 1356, 1361-62 (Fed. Cir. 2005) ("[I]t frequently happens that patentees surrender more through amendment than may have been absolutely necessary to avoid particular prior art . . . .").

C. "Large Block" / "Large Block Input/Output Transfer"

Construction: No construction necessary, or in the alternative "substantially larger than 4 kilobytes."

1. Support For Vertica's Claim Construction

The Court should not construe this term because the patent does not allow for a more precise definition of what constitutes a "large block." The specification states that a block of 4K is an example of a small block. (Ex. A, '229 patent, col. 10, ll. 12-13). The patent also states

that a block of 64K is **an example** of a large block. (*Id.* at col. 10, ll. 12-20). The patent does not differentiate the size at which a block becomes a big block. (*Id.* (emphasis added)). Although Vertica had originally proposed using the phrase "substantially larger than 4K," that construction is only slightly more precise than the claim language itself, and Vertica is content with the original claim language (without construction). The problem here is that the patent does not define the size of a large block. Accordingly, the Court should not attempt to add a degree of precision that the patent does not provide.

2. Critique of Sybase's Construction

Whether or not the Court decides to construe this term, it should **not** adopt Sybase's proposed construction because it is incorrect. Sybase assumes that 64K must be the **minimum** size for a large block, but there is nothing in the specification or claims that support this assumption. The most that can be said is that large blocks are larger than 4K (a small block). If Sybase had intended a precise definition of large blocks (e.g., 64K or larger) it could have expressly stated this – but it did not. The Court should not read a limitation into the claims based on a single example, when there is no indication in the specification or prosecution history that this example was intended to set the floor of what should be considered a "large block" in the claim. *See Phillips v. AWH Corp.*, 415 F.3d 1303, 1324-24 (Fed. Cir. 2005). There is no basis for saying the lower bounds of a large block is "64K."

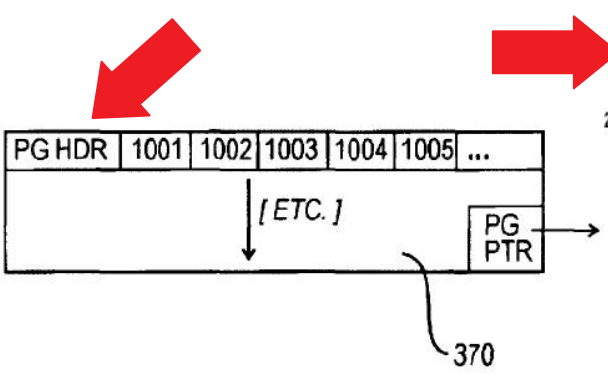
D. "Page Header"

Construction: Supplemental data at the beginning of a data page.[15]

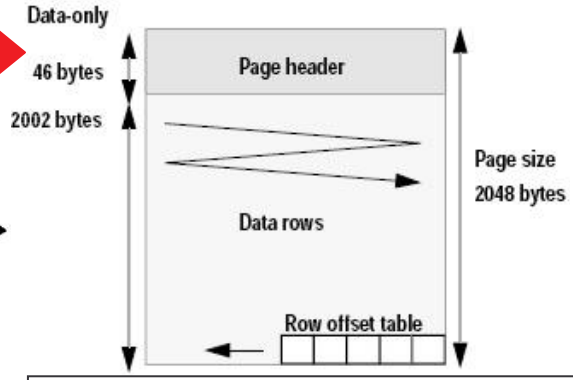
[15] This is modified from the "generic" definition Vertica originally proposed which used the term "block." Sybase objected, and so Vertica has changed the word "block" to "page."

1. Support For Vertica's Claim Construction

Figures 3A-3C illustrate the "page header" in the data page and at the beginning of the data page. The patent states that the page header includes a status flag indicating the page is a candidate for compression (Ex. A, '229 patent, col. 4, ll. 18-22), housekeeping information (*Id.* at col. 12, ll. 64-66), and page type (*Id.* at col. 14, ll. 42-44).



Ex. A, '229 patent, fig. 3B



Ex. D, Sybase Adaptive Server Performance and Tuning Guide, p. 3-4

A page header is at the beginning of the data page—it precedes the data. This is apparent from the use of the word "header," which refers to the top of a page. For example, a "header" in a Microsoft Word document refers to the information at the top of the page (as does Sybase's own "Performance and Tuning Guide." Technical dictionaries confirm this definition. (*See e.g.*, Ex. M, Microsoft Computer Dictionary (Microsoft Press 1998), pp.167-68 ("An information structure that precedes and identifies information that follows, such as a block of bytes in communications, a file on a disk, a set of records in a database, or an executable program")). This is what the figures of the patent show.

2. Critique of Sybase's Construction

Sybase's construction is incorrect because it does not require the supplemental data to be a part of the data page. The patent specification contradicts Sybase's proposed construction because it states that a page header is part of the data page: "the column-based **data page** 370

includes a **page header** (for housekeeping information) and a page pointer." (Ex. A, '229 patent, col. 13, ll.18-20 (emphasis added)). The '229 patent also states that when data is located on a page, the computed location must be **adjusted for any header information to the page**." (Ex. A, col. 4, ll. 8-13 (emphasis added)). The adjustment is necessary because the header information is at the beginning of the data page. A page header is not amorphous data that is "associated with" a data page. Rather, a page header refers to the supplemental information included within data page and which precedes the data.

E. Query Processing Means

Function: *Receiving a user query about data values stored in a particular column.*

Structure: *"Parser 261" using recursive descent parsing (column 7, ll. 45-46), plus the Find method and standard database "Column Operations" (column 45, ll. 9-17)*[16]

1. Support For Vertica's Claim Construction

Claim 20 requires a means for receiving a query and a means for executing the query: "The system of claim 16, further comprising: query processing means for receiving a user query about data values stored in a particular column, said query processing means including means for executing the query means by scanning only the data pages for the particular column." The receiving step is performed by the parser, which receives the query and converts the SQL statements into a format that can be used by the system. (Ex. A, '229 patent, col. 7, ll. 41-47). The patent describes the means for executing the query as the "find" method and "column operations" (such as AVG, RANGE, MIN, and MAX). These are the structures described in the patent related to these two functions.

[16] This construction is modified slightly from the original proposal to indicate the explicit structures disclosed in the '229 patent.

2. Critique of Sybase's Construction

Sybase's construction improperly seeks to require a number of other elements that are not required for these receiving or executing steps. For the query processing means limitation, Sybase identified only the "Engine 260" as the structure. However for the means for executing limitation, Sybase has expanded the alleged structure to require several components of the Engine, "**including** a Parser 261, Normalizer 263, Compiler 265, Execution Unit 269, Access Methods 270, and Buffer Managers 272." (Dkt. No. 107, Sybase's Claim Construction Brief, p. 38). Sybase's construction should be discarded because the "structure disclosed in the specification must be **clearly linked** to and capable of performing the function claimed by the means-plus-function limitation." *Default Proof Credit Card System, Inc. v. Home Depot U.S.A., Inc.*, 412 F.3d 1291, 1299 (Fed. Cir. 2005) (emphasis added). Sybase never identifies the necessary linking. Essentially, Sybase is pointing to the entirety of the database software and the corresponding structure, rather than the specific structures clearly linked to the claimed function.

As to the elements specified, Vertica agrees that the parser (which receives the message) and the access methods (which includes the find method and column operations) are relevant structures, but the rest of the alleged structure is superfluous. There is nothing in the patent that suggests these elements perform the receiving or execution functions. Moreover, were these components included, the claim would be indefinite because there is no disclosure of any structure regarding these components. These components are only disclosed in functional terms. For example, the following is the only description the patent has of the normalizer: "the Normalizer performs error checking, such as confirming that table names and column names which appear in the query are valid (e.g., are available and belong together). Finally the Normalizer can also look up any referential integrity constraints which exist and add those to the query." (Ex. A, '229, col. 7, ll. 50-55). The patent states what the normalizer does, but does not

explain how it does it. The Court should construe the claim to include only the structures that perform the receiving and executing functions.

F. On a Page-By-Page Basis

Construction: The compression type for each page is determined independently of the compression type for any other page.

1. Support For Vertica's Claim Construction

Claim 21 claims a system that is capable of specifying on a page-by-page basis which data compression type should be used: "means for compressing each data page according to a type of compression specified on a page-by-page basis." The ordinary meaning of the words "page-by-page" is that the compression type for each page is independent of the compression scheme of any other page. This ordinary meaning is confirmed by the specification: "The pages are further optimized for compression by storing in the page header a status flag indicating . . . what type of compression is best suited for the data on **that** page." (Ex. A, col. 4, ll. 18-22). The compression type "is settable on a page-by-page basis." (Ex. A, col. 4, ln. 22.) Claim 21 is narrow and expressly requires that the system be capable of setting different compression schemes for different pages within the same column.

2. Critique of Sybase's Construction

Sybase's proposed construction seeks to eliminate the page-by-page limitation and make it column-by-column. Vertica's system uses column-by-column compression (not page-by-page compression), so this is an attempt to broaden the patent to cover Vertica's system. It is true that the patent discloses both column-by-column compression and page-by-page compression, but the claim specifically calls for page-by-page compression.

G. Wherein said Type Of Compression Is Selected From [List]

Construction: No construction necessary. [17]

H. "Includes A Selective One Of [List of Items]

Construction: No construction necessary.

I. "Creating For Each Column . . . At Least One Associated Data Page For Storing Data Values For The Column"

Construction: No construction necessary.

J. "Data Values For Each Column . . . Are All Stored Together" / "Stores Together All Of The Data Values For A Particular Column"

Construction: No construction necessary, or in the alternative, "creating a group of one or more data pages per column for storing the data values for each column."

Upon review both parties' constructions, Vertica believes that the actual claim language is better than the offered constructions. In addition, Sybase's construction **cannot** be adopted because it is limited to a subset of data values, and not all data values. This limitation require **all** data values to be "stored together." If the claim were limited to **a single page**, then Sybase's construction might make sense, but the claim allows for more than one data page in a page chain. When there are two (or more) data pages, a system cannot store "all of the data" on a single page, as Sybase's construction suggests. If a construction is required, Vertica's proposal is more true to the claim language as a whole and to the '229 patent.

VI. INDEFINITE TERMS

There are six terms that are indefinite under 35 U.S.C. § 112, ¶ 2:

- linking together all of said at least one data page

[17] Vertica had originally contended that this term and the following two needed construction. However, upon reviewing Sybase's brief and comparing the parties' constructions to the actual claim language, Vertica now believes no construction is necessary.

- means for creating a vertical partition for each and every column of the database table, each vertical partition having data values for only a single column of the database table

- means for transferring each vertical partition to and from the storage device, so that at least some data values for a particular column are stored together at a contiguous location on the storage device.

- wherein said means for creating a vertical partition for each column includes: forming a vertical page chain for each column, said vertical page chain storing only those data value of the records which correspond to the column for the page chain.

- data compression means for compressing each data page according to a type of compression specified on a page-by-page basis.

- Large block transferring means for retrieving data values of a particular vertical partition as a single large block

Vertica has addressed these claim limitations in its contemporaneously-filed motion for summary judgment of indefiniteness. Rather than repeat the argument, Vertica hereby incorporates by reference its memorandum in support of that motion.

VII. CONCLUSION

For the foregoing reasons, Vertica respectfully request that the Court adopt its proposed constructions.

Dated: October 13, 2009

Respectfully submitted,

FISH & RICHARDSON P.C.

By: */s Steven R. Katz*

Steven R. Katz, BBO#642732
FISH & RICHARDSON P.C.
225 Franklin Street
Boston, MA 02110-2804
617-542-5070 (Telephone)
617-542-8906 (Facsimile)

Ralph Ritch Roberts III
State Bar No. TX 24041794
Roberts@fr.com
FISH & RICHARDSON P.C.
1717 Main St., Suite 5000
Dallas, TX 75201
(214) 747-5070 (Telephone)
(214) 747-2091 (Facsimile)
Counsel for
VERTICA SYSTEMS, INC.

CERTIFICATE OF SERVICE

The undersigned hereby certifies that a true and correct copy of the above and foregoing document has been served on all counsel of record who are deemed to have consented to electronic service via the Court's CM/ECF system per Local Rule CV-5(a)(3). Pursuant to the Court's Order, courtesy copies have been delivered to the Court via hand delivery and shipped to the Technical Advisor for overnight delivery.

/s Steven R. Katz
Steven R. Katz